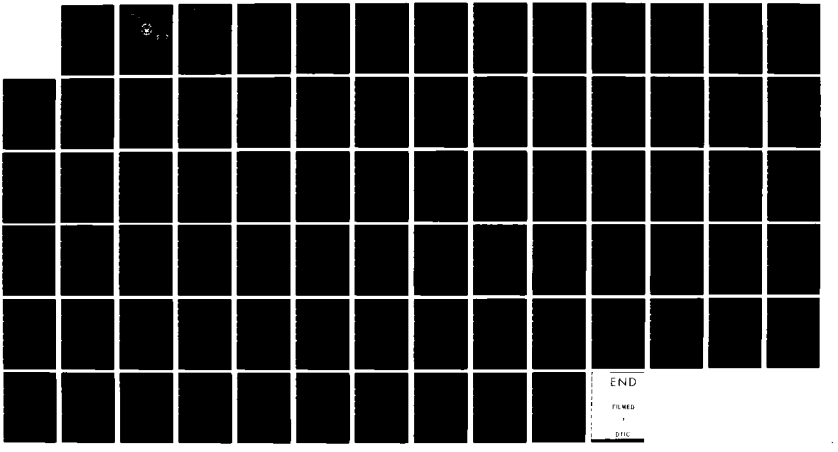
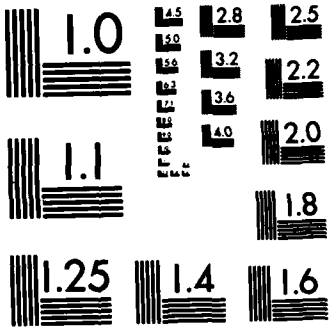
AD-A123 774 COMPUTATIONAL ADVANCES IN THE SOLUTION OF LARGE-SCALE SET COVERING AND SET PARTITIONING PROBLEMS(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA D O BAUSCH OCT 82 1/1

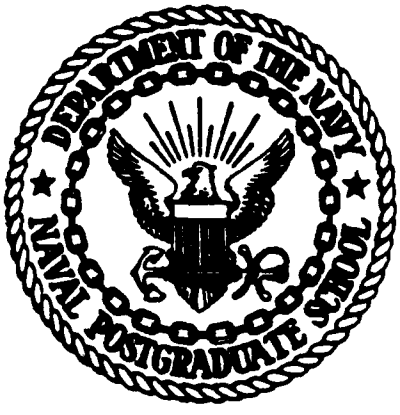
UNCLASSIFIED F/G 12/2 NL





(2)

NAVAL POSTGRADUATE SCHOOL
Monterey, California



THESIS

COMPUTATIONAL ADVANCES IN THE SOLUTION OF LARGE-SCALE SET COVERING AND SET PARTITIONING PROBLEMS

by

Dan O. Bausch

October 1982

Advisor: G. G. Brown

Approved for public release, distribution unlimited

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A123774
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle) Computational Advances in the Solution of Large-Scale Set Covering and Set Partitioning Problems	5. TYPE OF REPORT & PERIOD COVERED Master's Thesis October 1982
6. PERFORMING ORG. REPORT NUMBER	
7. AUTHOR(s) Dan O. Bausch	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940	12. REPORT DATE October 1982
	13. NUMBER OF PAGES 77
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report) Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release, distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Set covering, set partitioning, integer programming, branch and bound, generalized networks, processing networks, elastic programming, vehicle routing, tanker routing, airline crew scheduling, ship scheduling.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The Set Covering Problem (SCP) and the Set Partitioning Problem (SPP) represent an important class of all-binary (0-1) Integer Linear Programs (ILP). A review of the literature reveals extensive application of the SPP/SCP model to a wide set of practical problems. The basic model is explained, and then many of the actual applications of this powerful model discovered in the literature review are discussed. The problems derived from these applications

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

are difficult to solve with any method, and are particularly difficult to solve with optimal or exact algorithms. Various solution techniques are investigated within the framework of the classical simplex method with branch and bound enumeration. Several reformulations of the SPP/SCP as Integer Generalized Networks are examined. Extensive computational results are reported for several "real world" large-scale problems, and a convenient, compact format for data input is proposed as a standard for this problem class.

Approved for public release, distribution unlimited

Computational Advances in the Solution of Large-Scale Set Covering and Set Partitioning Problems

by

Dan O. Bausch
Captain, United States Marine Corps
B.A., Rice University, 1976

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the
NAVAL POSTGRADUATE SCHOOL
October, 1982

Author: Dan O. Bausch

Approved by: GERALD G. BROWN
Thesis Advisor

Richard D. McBride

Second Readers

Chairman, Department of Operations Research

W. M. Woods
Dean of Information and Policy Sciences

ABSTRACT

The Set Covering Problem (SCP) and the Set Partitioning Problem (SPP) represent an important class of all-binary (0-1) Integer Linear Programs (ILP). A review of the literature reveals extensive application of the SPP/SCP model to a wide set of practical problems. The basic model is explained, and then many of the actual applications of this powerful model discovered in the literature review are discussed. The problems derived from these applications are difficult to solve with any method, and are particularly difficult to solve with optimal or exact algorithms. Various solution techniques are investigated within the framework of the classical simplex method with branch and bound enumeration. Several reformulations of the SPP/SCP as Integer Generalized Networks are examined. Extensive computational results are reported for several "real world" large-scale problems, and a convenient, compact format for data input is proposed as a standard for this problem class.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENT

I would like to thank Professors Glenn W. Graves and Richard D. McBride for their copious technical assistance, Professors Roy E. Marsten and David Ronen for providing test problems, and David Norman for his indulgence in my use of CPU time. Lastly, I wish to express my appreciation to Professor Jerry Brown, without whose support this venture into the rarefied air of large-scale mathematical programming would not have been possible.

I. INTRODUCTION

The Set Covering Problem (SCP) and the Set Partitioning Problem (SPP) represent an important class of all-binary (0-1) Integer Linear Programs (ILP's). These problems have binary variables, binary constraint coefficients and unit or integer resources.

The basic SPP/SCP model has been known for over 25 years. It is enticing in formulation and deceptively simple. A review of the open literature reveals extensive application of the SPP/SCP model to a wide range of problems, including airline crew scheduling, vehicle routing, and facilities location. Even though the model has been intensively studied for both its intriguing binary structure and its potential for practical application, exact solution technologies for large-scale problems were not evident until the work of such researchers as Marsten [Ref. 1] began to appear in the early 1970's. Other early contributors are listed by Christofides [Ref. 2].

After first defining the basic model and discussing many of its applications, several reformulations of the SPP/SCP will be examined. Glover and Mulvey [Ref. 3] have presented two reformulations of the binary ILP as an Integer Generalized Network (GNIP). There is very little computational evidence in the literature concerning these reformulations; therefore, the computational behavior of this approach will be tested and results reported for several problems. Another reformulation of the SSP/SCP as an Integer Generalized Processing Network (a network with special side constraints) will be examined and its potential evaluated.

As indicated by the many reformulations, manipulative techniques, and heuristic methods appearing in the literature, these problems are difficult to solve reliably with any method, and are particularly difficult to solve with optimal or exact algorithms. Various solution techniques based on the classical simplex method with branch and bound enumeration are investigated in this study. Some of the techniques examined are basis factorization, elastic programming, enumeration schemes, network and linear programming relaxation, logical reduction, and heuristic methods for obtaining starting solutions. Extensive computational results are reported for several "real world" large-scale problems, and a convenient, compact format for data input is proposed as a standard for this problem class.

II. THE BASIC MODEL AND MODEL GENERATION

A. THE BASIC MODEL

The SCP formulated as an ILP is of the form:

$$(1) \quad \underset{j}{\text{MIN}} \quad \sum_{j=1}^{n} C_j X_j$$

$$(2) \quad \text{s.t.} \quad \sum_{j=1}^{n} a_{ij} X_j \geq b_i \qquad i = 1, \ldots, m$$

$$(3) \quad X_j \in \{0,1\} \qquad j = 1, \ldots, n$$

$$(4) \quad C_j \geq 0 \qquad j = 1, \ldots, n$$

$$(5) \quad b_i \geq 0 \text{ and integer } i = 1, \ldots, m$$

$$(6) \quad a_{ij} = \begin{cases} 1 \text{ if column } j \text{ covers row } i, \\ 0 \text{ otherwise.} \end{cases}$$

A minimal cost set of columns must be selected from X_j such that the magnitudes of the right-hand sides (RHS), b_i, are "covered" or satisfied. If (2) and (5) are replaced by

$$(7) \quad \sum_{j=1}^{n} a_{ij} X_j = 1, \qquad i = 1, \ldots, m,$$

we have the SPP (sometimes referred to in the literature as the equality constrained SCP). This restriction of the SCP exhibits sufficient modelling and computational interest to be studied in its own right. For the SPP, the rows {i} represent a set which must be partitioned by a combination of mutually exclusive columns at minimum cost.

B. SIDE CONDITIONS

Many practical applications of the SPP/SCP formulation add logical constraints to the basic model discussed above. For example, suppose there are p sets of columns S_k, k = 1, ..., p in the model and only one column from each of the sets S_k is eligible to be included in the final solution. This restriction will produce constraints

$$(8) \quad \sum_{j \in S_k} X_j = 1 \text{ for all } k.$$

In another case, suppose that the solution must include exactly J columns. This results in the cardinality constraint

$$(9) \quad \sum_{j} X_j = J$$

being appended to the basic model. Introductory modelling texts such as Wagner [Ref. 4] and Gaver and Thompson [Ref. 5] discuss many such logical conditions formulated with binary variables. Any or all of these logical conditions can be included to extend the basic model for the purpose required.

C. COLUMN GENERATION

The art of formulating the practical SPP/SCP lies in the schemes used for column generation. It is possible, of course, to generate all $2^m - 1$ columns capable of covering or partitioning the rows, but for any relatively large number of rows, the problem becomes intractable. This "all possible combinations" formulation is known as the Complete SPP/SCP, and even though techniques are emerging for attacking such problems [Ref. 6], efforts must be made to keep the number of permissible columns within the capabilities of the optimizer being used. Editing reductions can be

realized by incorporating such conditions as managerial specifications of operating policy; dimensional restrictions on time, distance, and space; legal restrictions; labor union restrictions; cash flow restrictions; environmental restrictions; and as many other "real world" constraints as can be included in the column generation process.

Incorporating such conditions into the column generator can handle most, if not all, side conditions and feasibility issues without including them as extensions of the basic SPP/SCP model. Some examples are described by Marsten and Muller [Ref. 7], Shanker, Turner, and Zoltners [Ref. 8], and Cullen, Jarvis, and Ratliff [Ref. 9].

D. THE OBJECTIVE FUNCTION

The cost coefficients C_j for the basic model can be of two types: physical and ordinal. A physical cost is a coefficient in units of dollars, miles, time, etc., and represents the cost of covering certain rows with column j. The associated physical objective function expresses the cost of covering or partitioning the set represented by the rows.

It is quite often the case, though, that the cost coefficients serve only as a means of distinguishing between alternate columns. In many political and social models, for example, a column will be assigned a subjective number depicting some measure of acceptability (or unacceptability) thus effecting an ordinal ranking structure in the objective function. The objective then becomes a matter of selecting those columns which minimize the ordinality. A much-used special case of the ordinal cost structure is the unit-cost objective function in which $C_j = 1$ for all j. The optimal solution for the unit-cost objective function is the

minimum number of columns capable of covering or partitioning the row set without regard to physical cost or ordinal ranking.

III. APPLICATIONS AND PROBLEM DESCRIPTIONS

A. APPLICATIONS

Set Covering and Set Partitioning Problems have been studied extensively because of their many practical applications. The surveys by Garfinkel and Nemhauser [Ref. 10] and Balas and Padberg [Ref. 11] list many useful applications which have appeared in the literature. Some of these are listed below, along with a few which have subsequently appeared.

	APPLICATION	REFERENCE (*Unsighted)
1.	Truck Deliveries	[Ref. 12],* [Ref. 13],* [Ref. 14]*, [Ref. 15], [Ref. 16]*.
2.	Tanker Routing	[Ref. 17].
3.	Aircrew Scheduling	[Ref. 18],* [Ref. 19],* [Ref. 20],* [Ref. 21],* [Ref. 22]. [Ref. 23], [Ref. 7].
4.	Facilities Location	[Ref. 24], [Ref. 25],* [Ref. 26], [Ref. 27], [Ref. 28].
5.	Air Fleet Scheduling	[Ref. 29],* [Ref. 7].
6.	List Selection	[Ref. 30],
7.	Political Districting	[Ref. 31],* [Ref. 32],* [Ref. 33].
8.	Nuclear and Conventional Targeting	(See Appendix A).
9.	Information Retrieval	[Ref. 34],* [Ref. 35].*
10.	Symbolic Logic	[Ref. 36].*
11.	Switching Theory	[Ref. 37], [Ref. 38],* [Ref. 39],* [Ref. 40].*
12.	Stock Cutting or Trimming	[Ref. 41].*
13.	Line Balancing	[Ref. 42].*

APPLICATION	REFERENCE
14. Capacity Balancing	[Ref. 43].*
15. PERT-CPM	[Ref. 36].*
16. Frequency Allocation	[Ref. 44].
17. Tracking Problems	[Ref. 45].
18. Vehicle Routing	[Ref. 9].
19. Sales Territory Design	[Ref. 8].
20. Coloring Problems	[Ref. 46],* [Ref. 47], [Ref. 48].*
21. Disconnecting Paths in a Graph	[Ref. 49], [Ref. 50].
22. Cyclic Scheduling Problems	[Ref. 51], [Ref. 52].

B. THE TRUCK DELIVERY PROBLEM

The first application we will examine is one that appears quite often in textbooks and is a simple illustration of the basic model. This problem will also provide an example which will be carried forward through discussion in later sections.

Consider the problem of making deliveries to m locations by truck (rail, aircraft, ship, messenger, etc.). There are n feasible routes to choose from and $a_{ij} = 1$ if location i is on route j. A cost C_j (say, time, dollars, miles) is assigned to route j. An optimal partition gives a minimal cost routing that makes each delivery exactly once. An optimal cover gives a minimal cost routing that makes sufficient deliveries to satisfy the demand at each location. The optimal solution to the unit-cost problem yields the minimum number of trucks necessary to make the required deliveries.

Table 1 is the explicit tableau for an illustrative example of the SPP. The flight scheduler for a small West Coast air freight company has

TABLE 1. AIR FREIGHT EXAMPLE

	R1	R2	R3	R4	R5	R6	R7	RHS
Los Angeles	1	0	0	0	0	0	0	= 1
San Francisco	1	1	1	0	0	0	0	= 1
San Jose	1	1	1	0	0	0	0	= 1
Denver	0	0	1	1	1	0	0	= 1
Portland	0	0	0	1	1	0	0	= 1
Seattle	0	0	0	1	1	1	0	= 1
San Diego	0	0	0	0	1	1	1	= 1
Costs	0	0	0	0	6	7	4	obj.

been assigned the task of delivering exactly one of seven identical packages to each of seven western cities by tomorrow morning. All of the delivery points can be reached in the required time with the current schedule except San Diego. There are only three feasible ways to make the San Diego delivery: extend route 5, extend route 6, or add a new route 7. The cost of the alternatives is calculated and appears in the tableau. By inspection, there are only two feasible solutions: (1) Routes 1 and 5 at a cost of 6, and (2) Routes 1, 4, and 7 at a cost of 4. The minimum cost solution, therefore, is to add flight 7 to the current schedule. The unit-cost solution or minimum partition is to use solution (1).

Two large-scale, real-life problems of this type have been examined in this study: TRUCK and TANKER. TRUCK is a nationwide, intercity truck routing problem, a large SCP, and fits the basic description above.

TANKER is a worldwide oil tanker fleet scheduling problem which extends the basic SPP model to help choose between company-owned and charter tankers to meet refinery delivery requirements from available loading volumes and origins. Each cargo, company-owned ship, and potential charter vessel is represented by a row. Cargoes must be carried, and ships must either be used, or scheduled in demurrage. Each column represents a feasible route for a particular ship; during the planning horizon, it may carry zero, or more cargoes. The cost of each route may be calculated ordinally (based on fleet size) or economically (based on operating costs).

The problem dimensions for TRUCK and TANKER are listed in Table 2. NZEL is the total number of non-zero elements, and NCE is the average number of non-zero elements per column.

TABLE 2. TRUCK DELIVERY PROBLEM DIMENSIONS

	ROWS	COLUMNS	NZEL	NCE	MODEL
TRUCK	239	4752	30075	8.0	SCP
TANKER	166	7563	31289	4.1	SPP

C. THE AIRCREW SCHEDULING PROBLEM

An airline has a set of m flight legs, each of which requires a crew. Given the airline's timetable, a set of n possible crew rotations can be generated. Each crew rotation is a sequence of scheduled flight segments constituting a roundtrip (a sequence departing from and returning to one of the airline's crew bases). A cost is calculated for each rotation,

and once a complete set of flyable rotations has been generated, the problem is to select an optimal, feasible subset.

Depending on whether or not crew members are allowed to be passengers on certain flights, optimal covers or optimal partitions yield optimal schedules. "Deadheading" is the practice of allowing a crew to travel as passengers on certain flights. Planned deadheading can be accommodated with the partitioning model. If rotation j concludes a planned deadhead on flight segment i, then a_{ij} is set to zero rather than one. If unplanned deadheading is allowed, however, then a covering problem must be solved.

A typical side condition common to these models is the Crew Base Constraint of the form

$$\sum_{j \in D_s} H_j X_j = M_s \text{ for } s = 1, \ldots, S$$

where H_j is the number of flying hours, per month, associated with rotation j; M_s is the maximum number of flying hours available per month at crew base s; and D_s is the set of rotations flown out of crew base s.

All of the problems of this type were provided by Professor Roy E. Marsten, University of Arizona, and are described in [Ref. 23]. TIGER1 and TIGER2 are examples of crew scheduling problems generated for Flying Tiger Airlines. AMERICAN is a large crew scheduling problem generated by American Airlines. (All of the airline problems in this study were solved without crew base constraints.) The last problem in this class is BUS, a driver-scheduling problem generated for Helsinki City Transport as described in [Ref. 23].

TABLE 3. AIRLINE CREW SCHEDULING PROBLEM DIMENSIONS

	ROWS	COLUMNS	NZEL	NCE	MODEL
TIGER1	160	636	4134	6.5	SPP
TIGER2	107	2188	8266	3.8	SPP
AMERICAN	95	9318	57293	6.1	SPP
BUS	56	530	3365	6.3	SPP

D. THE MAXIMAL SET COVERING PROBLEM

Either the facilities location problem or the list selection problem can be formulated as a Maximal Set Covering Problem. This problem differs from the basic SCP because we no longer require that all rows be covered, rather the objective is to cover as many rows as possible subject to various constraints. To accomplish this, m continuous variables, Y_i, are added to the basic model to produce the following Mixed Integer Program (MIP):

$$\underset{i}{\text{Min}} \quad \sum_{i=1}^{m} Y_i$$

$$\text{S.T.} \quad \sum_{j=1}^{n} a_{ij}X_j + Y_i \geq 1 \qquad i = 1, \ldots, m$$

$$\sum_{j=1}^{n} X_j = J \tag{9}$$

$$\sum_{j=1}^{n} C_jX_j \leq B \tag{10}$$

$$0 \leq Y_i \leq 1 \qquad i = 1, \ldots, m$$

$$X_j \in \{0,1\} \qquad j = 1, \ldots, n.$$

The constraint (9) limits the number of rows which may be covered by specifying that only J columns can be used. The constraint (10) is a budget constraint which specifies that as many rows as possible be covered for B dollars. The sense of the objective function here is to minimize the number of rows left uncovered.

Another formulation in the same spirit replaces the objective function by the familiar Min $\sum_j C_j X_j$, and adds the constraint

$$(11) \quad \sum_{i=1}^{m} Y_i \leq M.$$

This formulation seeks the minimum cost set of columns which leaves at most M rows uncovered.

Dwyer and Evans [Ref. 30] have applied a similar formulation to the "list selection problem." The list selection problem selects a set of subscriber lists which maximizes the proportion of customers reachable with direct mail pieces. The rows correspond to magazine subscribers, and the columns to individual magazines. Let $a_{ij} = 1$ if individual i subscribes to magazine j, and zero otherwise.

Moore and Revelle [Ref. 28] have applied this formulation to a hierarchical facilities location problem. The rows represent demand points and the columns represent various location strategies. The objective is to pick those strategies which cover as much of the demand as possible.

STEINER1 and STEINER2 are two computationally difficult set covering problems published by Fulkerson, Nemhauser, and Trotter [Ref. 53]. Each row has exactly three non-zero elements, $b_i = 1$ for all i, and the objective function is of the unit-cost type. These basic SCP's were

extended to MCOVER1 and MCOVER2 respectively, in order to evaluate the difficulty of the Maximal Covering Problem.

TABLE 4. MAXIMAL COVERING PROBLEM DIMENSIONS

	ROWS	COLUMNS	NZEL	NCE	MODEL
STEINER1	117	27	351	13	SCP
MCOVER1	118	144	495	13	SCP(ext)
STEINER2	330	45	990	22	SCP
MCOVER2	331	375	1365	22	SCP(ext)

Only the budget constraint (10) was added to produce the extended problems. The value of B was set so that MCOVER1 seeks the same optimal solution as STEINER1, and MCOVER2 seeks the same optimal solution as STEINER2.

IV. COMPUTATIONAL DIFFICULTIES

A. THE BRANCH AND BOUND ENUMERATION METHOD

It is evident that the SCP/SPP is a powerful model with many useful applications. Unfortunately, it is also true that the large-scale SCP/SPP is difficult to solve to optimality. In fact, Karp [Ref. 54] has shown the set covering problem to be an NP-hard combinatorial problem. The solution techniques investigated here involve simplex-based enumeration, often called branch and bound.

Branch and bound is an enumerative method that has been used successfully to optimize a variety of combinatorial problems. The basic principle is to methodically search the set of possible integer solutions in such a way that not all possibilities need be explicitly considered. The theoretical framework for this study is provided in the following, which has been adapted from Geoffrion and Marsten [Ref. 10]. The procedure of branch and bound is described in terms of three concepts: separation, relaxation, and fathoming.

1. Separation

For any optimization problem (P), let F(P) denote its set of feasible solutions. Problem (P) is said to be separated into subproblems if the following conditions hold:

1. Every feasible solution of (P) is a feasible solution of exactly one of the subproblems.
2. A feasible solution of any of the subproblems is a solution of (P).

The procedure is to first make a reasonable effort to solve (P). If this effort is unsuccessful, separate (P) into two subproblems, thereby initiating what will be called a candidate list of subproblems. A reasonable representation of the candidate list may be an "enumeration tree" which reveals the partial ordering of consideration among candidates in the list. Extract one of the subproblems from the list and call it the current candidate problem (CP). If (CP) can be solved, extract a new candidate problem from the list; otherwise, separate (CP) and add its "descendants" to the candidate list. Continue in this fashion until the candidate list is exhausted (i.e., every branch of the enumeration tree has been examined). If we refer to the best solution found so far to any candidate problem as the current incumbent, then the final incumbent must obviously be an optimal solution of (P).

The technique of separation involves "branching" on a single integer variable. For the SPP/SCP where X_j is declared to be a binary variable, the ILP can be separated into two subproblems by means of the mutually exclusive and exhaustive restrictions $X_j = 0$ or $X_j = 1$. An enumeration tree may be visualized with a vertex associated with each separation and an edge with each restriction. The tree predecessor relationship among vertices reveals the ordering among separations and their associated restrictions. This enumeration tree provides a visually appealing illustration of the solution sequence.

2. Relaxation

Any constrained optimization problem (P) can be "relaxed" by loosening its constraints, resulting in a new problem (P_R). By far the most popular type of relaxation for the ILP is to replace the integrality

restriction on the variables of (P) by simple bounds on the variables, producing the continuous problem (P_R). The only requirement for (P_R) to be a valid relaxation is that $F(P) \subseteq F(P_R)$. For the minimization problem, this definition implies:

1. If (P_R) has no feasible solutions, then the same is true of (P).
2. The minimal value of (P) is no less than the minimal value of (P_R).
3. If an optimal solution of (P_R) is feasible in (P), then it is an optimal solution of (P).

In selecting between the alternative types of relaxation for a given problem, there are two main criteria to be considered. First, it is desirable for the relaxed problem to be significantly easier to solve than the original. Second, one would like (P_R) to yield an optimal solution of (P), or, failing that, the minimal value of (P_R) should be as close as possible to that of (P). The distance between the minimal values of (P_R) and (P) is often described as the "gap," and is used as a measure of the "strength" (small gap) or "weakness" (large gap) of the relaxation (P_R). Unfortunately, the objectives that (P_R) be both "strong" and easy to solve are antagonistic. In general, the easier (P_R) is to solve, the greater the "gap" is between the original and relaxed problems.

3. Fathoming

Let (CP) be a typical candidate problem arising from the attempt to solve (P). The ultimate objective in dealing with (CP) is to determine whether its feasible region F(CP) may possibly contain an optimal solution of (P), and if it does, to find it. If it can be ascertained by some means that F(CP) cannot contain a feasible solution better than the incumbent (the best feasible solution yet found), this is certainly

good enough to dismiss (CP) from further consideration, and we say that (CP) has been fathomed. If an optimal solution of (CP) can actually be found, we also say that (CP) has been fathomed. In either case, the candidate problem has been entirely resolved for purposes of enumeration, and no further separations of (CP) are necessary. Thus, the subproblems which would arise as restricted descendants of (CP) have been enumerated implicitly by either the bounding argument or the feasibility argument.

Candidate problem (CP) is fathomed if any one of these criteria is satisfied:

1. An analysis of (CP_R) reveals that (CP) has no feasible solution.
2. An analysis of (CP_R) reveals that (CP) has no feasible solution better than the incumbent.
3. An analysis of (CP_R) reveals an optimal solution of (CP); e.g., an optimal solution of (CP_R) is found which happens to be feasible in (CP).

4. General Tree Search Procedure

STEP 1: Initialize the candidate list with the ILP. Set the incumbent value, Z*, equal to infinity.

STEP 2: STOP if the candidate list is empty. If there exists an incumbent then it must be optimal in the ILP, otherwise ILP has no feasible solution.

STEP 3: Select a candidate problem (CP) from the list and solve its relaxation (CP_R).

STEP 4: Fathoming Criterion 1. If the outcome of STEP 3 reveals (CP) to be infeasible, go to STEP 2.

STEP 5: Fathoming Criterion 2. If the outcome of STEP 3 reveals (CP) has no feasible solution better than the incumbent, Z*, go to STEP 2.

STEP 6: Fathoming Criterion 3. If the outcome of STEP 3 reveals an optimal solution of (CP), go to STEP 8.

STEP 7: Separate (CP) and add its descendants to the candidate list. Go to STEP 2.

STEP 8: A feasible solution of ILP has been found. If the value of the (CP) is less than Z*, record this solution as the new incumbent and set Z* = value of (CP). Go to STEP 2.

The degrees of freedom in STEP 3 provide a host of options. Critical among these is the selection mechanism for branch variables. A good branching strategy makes it possible to avoid searching large portions of the enumeration tree, thus greatly reducing time spent in the enumeration process. STEP 3 can also be prohibitively expensive if the solution of (CP_R) is not easy to generate from the solution of (CP). This step requires either storage of many (CP) solutions or a restriction in the sequence for branch solutions. For instance, "fixed order enumeration" permits branching only on the last element in the candidate list previously associated with a (CP).

Most successful implementations of this general scheme use the solution of the LP relaxation of the ILP to obtain the bounds required for the branch and bound enumeration. Christofides [Ref. 2] and Marsten [Ref. 1] report that most of the current large-scale algorithms use LP to obtain bounds for their various enumeration procedures. Exceptions are Etcheberry [Ref. 55], who uses "Lagrangian Relaxation" to obtain the bounds, and Glover and Mulvey [Ref. 3], who reformulate and use Generalized Network Relaxations.

The success of the branch and bound scheme depends on good branching strategies and the ability to obtain good bounds efficiently during the tree search. Typically, many LP's must be solved and even though the integer requirement is relaxed for each LP restriction, there are two serious problems associated with these LP's which make them hard to solve: numerical instability and degeneracy.

B. NUMERICAL INSTABILITY

The concept of a basis for the linear program must first be discussed in order to explain the computational difficulties. Consider the system of equalities AX = b where X is an n-vector, b an m-vector, and A is an m x n matrix ($m \leq n$). From the n columns of A, we select a set of m linearly independent columns and denote the m x m matrix determined by these columns by B. The matrix B is then non-singular and we may uniquely solve the equations $BX_B = b$ for the m-vector X_B, namely, $X_B = B^{-1}b$. If all $n - m$ components of X not associated with columns of B are set to equal zero, the solution to the resulting set of equations is said to be a basic solution to AX = b with respect to the basis B. B is called a basis since its m linearly independent columns span the space E^m.

It can be seen from the above explanation that the transformation by the basis inverse is necessary to obtain a basic solution. It is also true that all large-scale LP systems available today require some form of representation of this basis inverse transformation in order to function efficiently. One popular representation is the Product Form of the Inverse described by Orchard-Hays [Ref. 56]. Another example is the explicit sub-kernel representation described by Graves [Ref. 57].

Unfortunately, the columns of the SPP/SCP are often *nearly* linearly dependent. For instance, a route generator will produce a base route to, say, five locations. By substituting alternate locations one at a time into the base route, many routes are generated which differ by only one or two elements. This can produce an ill-conditioned basis whose inverse can contain numbers so large, or so small that after a few iterations with real arithmetic, the computer is unable to maintain sufficient

significance to provide the numerical stability necessary for the LP algorithm to converge, or if it does converge, to produce the true optimum.

To attempt to overcome the numerical instability, it is necessary to "clean up" the representation of the inverse by a process known as reinversion. There are many different reinversion schemes available, but in essence, they all use the original problem data to generate a new representation of the inverse which is relatively free from accumulated round-off error. Reinversion is computationally expensive, and for the SPP/SCP it is often necessary to reinvert quite frequently, thus slowing the computation of the bounds needed by the enumeration scheme.

C. DEGENERACY

The primal simplex algorithm for the solution of the LP proceeds from one basic feasible solution of the constraint set of a problem to another in such a way as to continually decrease the value of the objective function until a minimum is reached. If one or more of the basic variables in a basic solution has value zero, that solution is said to be a degenerate basic solution. For the SPP/SCP, it is important to note that seeking the minimum number of columns capable of covering or partitioning the row set is exactly equivalent to *maximizing* the primal degeneracy present in the optimal basic solution.

"Pivoting" is the name applied to the procedure which accomplishes a basis exchange. A degenerate basis exchange is one in which a column leaves the basis, a new column enters the basis, and the value of the objective function does not change. A degenerate pivot, then, exhibits the undesirable property that the basis exchange uses up computation time,

but no overt improvement in the objective function is realized. (We will ignore for the moment that we face a serious theoretical dilemma in demonstrating that the simplex method is finite in the presence of degeneracy.) Every pivot involves the update of the basis inverse representation; therefore, each update usually introduces round-off error. As discussed earlier, the basis inverse transformation for an ill-conditioned basis accumulates round-off error very quickly. In the presence of massive degeneracy, then, it is possible for the convergence of the primal simplex algorithm to be prohibitively slow, because an excessive amount of time is spent making degenerate basis exchanges and performing reinversion.

Degeneracy and round-off error can also produce a very serious phenomenon called "cycling." It is possible that a repeating sequence of degenerate basic solutions will be generated such that the simplex algorithm cycles endlessly without making progress. Most LP systems ignore the threat of cycling, because the repeating sequence is usually broken after reinversion "randomly" permutes the row order, thus evoking a new solution trajectory. If, however, significant time is spent in a cycle while waiting for reinversion to be triggered by the pivot count, the internal clock, or by a check on the rounding error, rapid solution of the LP will not be possible.

It has been determined that degeneracy and consequent cycling are significant obstacles for the efficient solution of the LP relaxation of the SPP/SCP with most of the available LP systems. Massive primal degeneracy is present as a consequence of the binary coefficients and the fact that for most SPP/SCP's, the right-hand sides of each row are

identical. It is this massive primal degeneracy which led Marsten to suggest the use of a dual algorithm for the solution of the LP [Ref. 1]. For the unit-cost objective function, a similar dual degeneracy can also be present. Although in most problems with general costs, the dual degeneracy is less severe than the primal degeneracy, even objective functions with varying costs can produce an effective degeneracy due to round-off error. The problem called TRUCK exhibits these characteristics.

V. REFORMULATIONS

A. THE NEED FOR REFORMULATION

The LP relaxation of the SPP/SCP can be numerically troublesome. One way to avoid this difficulty is to seek another relaxation which may be easier to solve. The alternate relaxations examined here are based on networks. Reformulation of the SPP/SCP as a network makes it possible to exploit an efficient solution technology. Network codes such as GNET [Ref. 58], and GENNET [Ref. 59] use basis handling procedures which require very little real arithmetic, thus avoiding much of the round-off error problem. Reformulation comes at the cost of making the problem larger, but it is hoped that the superior speed and numerical stability of the network approach will more than make up for the increase in problem size.

B. THE FIRST GENERALIZED NETWORK REFORMULATION

Glover, Hultz, Klingman, and Stutz [Ref. 60] have offered an interesting reformulation of any all-binary integer problem as an integer generalized network. The Generalized Network formulation is attractive because of the emergence of some very fast computer codes for solving generalized networks. Glover, et al., report that their code, NetG, is up to 50 times faster than state-of-the-art commercial LP codes on continuous network problems. GENNET has proved to be comparable in solution speed for the same class of problems. This reformulation, then, (which we will call GNIP-1) offers some promise for the solution of the SPP/SCP. It also

provides a way of describing the SPP/SCP in network terms which can make it easier to formulate and explain the model.

The SPP

$$(1) \quad \underset{j}{\text{MIN}} \quad \sum_{j=1}^{n} C_j X_j$$

$$(2) \quad \text{s.t.} \quad \sum_{j=1}^{n} a_{ij} X_j = 1 \qquad i = 1, \ldots, m$$

$$(3) \quad X_j \in \{0,1\} \qquad j = 1, \ldots, n$$

$$(4) \quad C_j \geq 0 \qquad j = 1, \ldots, n$$

$$(5) \quad a_{ij} = \begin{cases} 1 \text{ if column } j \text{ covers row } i, \\ 0 \text{ otherwise} \end{cases}$$

reformulated as a Generalized Network becomes the GNIP-1

$$\text{MIN} \quad \sum_{k} C_k X_k$$

$$\text{S.T.} \quad \sum_{k} X_k \geq 0$$

$$\sum_{k:\text{head } j} - M_k X_k + \sum_{k:\text{tail } j} Y_k = 0 \qquad j = 1, \ldots, n$$

$$(12) \quad \sum_{k:\text{head } i} Y_k = 1 \qquad i = 1, \ldots, m$$

$$X_k \in \{0,1\}$$

$$0 \leq Y_k \leq 1 \ .$$

For the SCP, (12) would be replaced by $\sum_{k:\text{head } i} Y_k \geq b_i$.

The procedure for drawing the network flow diagram is given below.

Given a SPP(SCP) with m rows, n columns, and NCE(j) non-zero elements per column,

1. Create a node i for each constraint, i = 1, ..., m; and give each node a demand of 1 (SCP $\geq b_i$).

2. Create a node j for each variable, j = 1, ..., n; where demand = supply = 0.

3. Create a super source node S and give it a supply ≥ 0.

4. Create a generalized arc X_k (S,j) for each original variable, k = 1, ..., n.

   a. Assign arc X_k a cost of C_j.
   b. Designate arc X_k as an integer (0 - 1) arc.
   c. Give arc X_k a multiplier M_k equal to NCE(j).

5. Create a pure network arc Y_k (j,i) for each non-zero element in column j.

   a. Assign arc Y_k a cost of zero.
   b. Assign arc Y_k an upper bound of one and a lower bound of zero.

The GNIP-1 reformulation of the Air Freight Example Problem is displayed in Figure 1.

It can be seen in Figure 1 that if the flow on a generalized arc $X_{k:\text{head } j}$ is zero, the flows on the continuous arcs Y_k emanating from the variable node j are also zero. It is also clear that if the flow on the generalized arc is 1, a flow of M_k arrives at the variable node, forcing a flow of 1 on each continuous arc incident to that node.

The telling disadvantage of the GN reformulation is the weakness of its continuous relaxation. When the integer restriction is removed for the arcs X_k, there is no assurance that an integral flow will arrive at the variable node. So far, this is comparable to the LP relaxation of the integer variable X_j. The difference between the LP relaxation and GN relaxation lies in the continuous arcs Y_k emanating from the variable node. Given a fractional supply, there is no assurance that the flows on each continuous arc will be the same. If the flows were identical, the

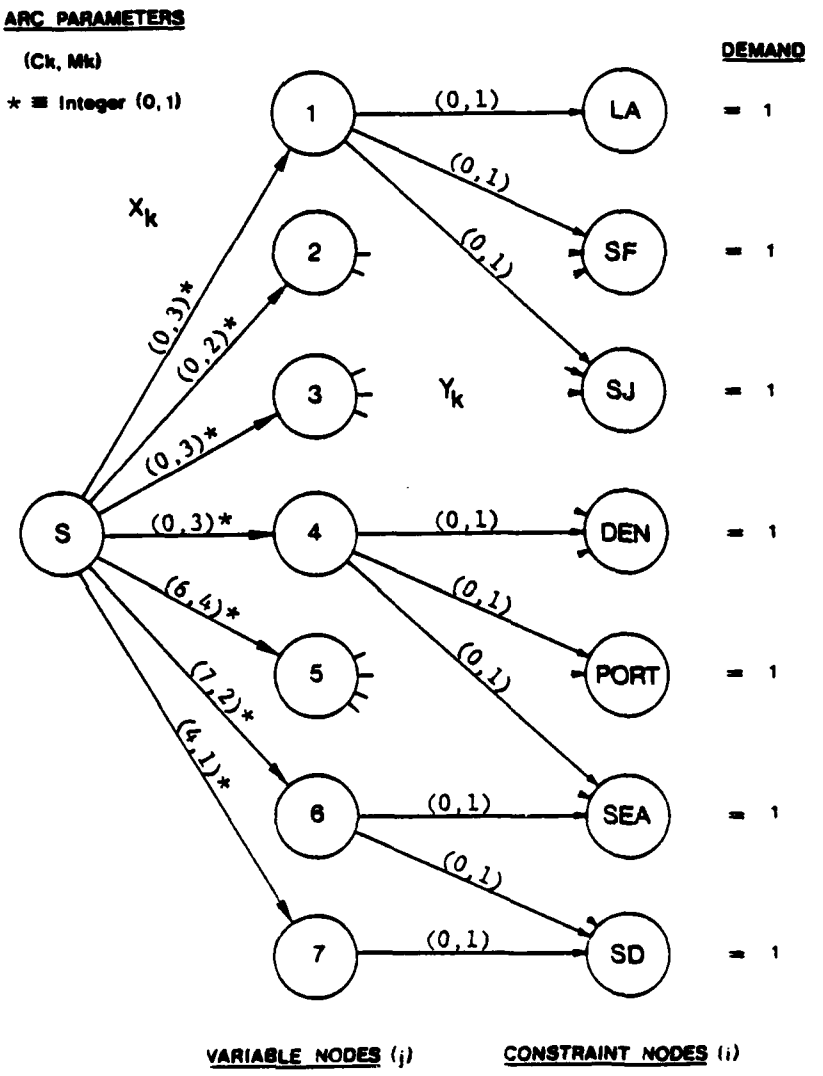


Figure 1. GNIP-1 Reformulation of the Air Freight Example

GN relaxation would be the same as the LP relaxation. Unfortunately, empirical evidence suggests that the flows differ widely. Furthermore, the problem increases as a function of the number of non-zero elements per column in the original ILP. Srinivasan and Thompson [Ref. 61] report a practical upper bound of three or four non-zeros per column for a similar reformulation of set partitioning problems.

Glover and Mulvey [Ref. 3] state that it is legitimate to manipulate the costs incident to a given variable node provided that these costs always sum to C_j. This can be interpreted as a form of "Lagrangian" manipulation, taking side constraints into the objective function, where these side constraints stipulate that the flow on each pure network arc incident to each variable node be the same. By linear programming duality, there exists some such assignment of costs for which the optimum objective function value for the GN is the same as that for the LP relaxation. Obviously, the trick is to find an exact or heuristic procedure of assigning these costs to "strengthen" the GN relaxation. Several attempts were made to distribute costs according to the proportion of flows on the first and subsequent GN relaxations, but these attempts proved ineffective.

C. THE SECOND GENERALIZED NETWORK REFORMULATION

One way to strengthen the GN relaxation is to reduce the number of continuous arcs in the reformulation. Glover and Mulvey [Ref. 3] have presented another GN formulation for the ILP (GNIP-2) which eliminates the super source node, the n generalized arcs emanating from it, and one continuous arc per variable node. For the ILP with m rows, n columns, and NCE(j) non-zero elements per column,

1. Create a node i for each constraint, i = 1, ..., m; and give each a supply of one (SCP $\geq b_i$).

2. Create a node j for each variable, j = 1, ..., n; where demand = supply = 0.

3. Create an arc (i,j) for each non-zero element in the ILP, connecting each constraint node to the appropriate variable node.

   a. Select one arc for each j and designate it as an integer (0-1) generalized arc X_k.

      (1) Assign arc X_k a cost of C_j.
      (2) Assign arc X_k a multiplier of M_k = NCE(j) - 1 (if NCE is greater than one).

   b. Designate the remaining arcs as continuous generalized arcs Y_k.

      (1) Assign arc Y_k an upper bound of 1.
      (2) Assign arc Y_k a cost of zero.
      (3) Assign arc Y_k a multiplier of $M_k = -1$.

   c. If NCE(j) = 1

      (1) Create a slack node S with a supply $\leq M$.
      (2) Create a *continuous* arc Y_k (S,j) as in 3b above.

The GNIP-2 reformulation of the Air Freight Example is displayed in Figure 2.

The above procedure produces the following mathematical program GNIP-2:

$$\text{MIN} \quad \sum_k C_k X_k$$

$$\text{S.T.} \quad \sum_{k:\text{head } j} M_k X_k + \sum_{k:\text{head } j} - Y_k = 0, \quad j = 1, \ldots, n;$$

$$\sum_{k:\text{tail } i} X_k + \sum_{k:\text{tail } i} Y_k = 1, \quad i = 1, \ldots, m; \tag{13}$$

$$X_k \in \{0,1\}$$

$$0 \leq Y_k \leq 1$$

For the SCP, (13) would be replaced by $\sum_{k:\text{tail } i} Y_k + \sum_{k:\text{tail } i} + Y_k \geq b_i$.

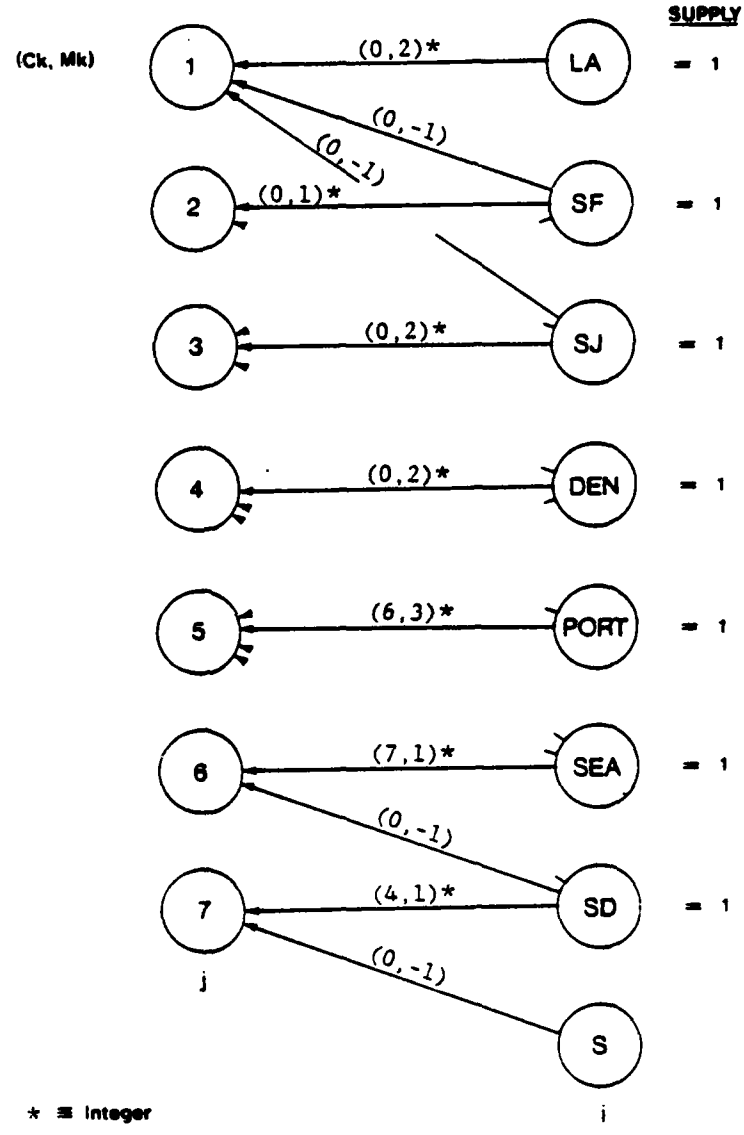


Figure 2. GNIP-2 Reformulation of the Air Freight Example

Table 5 shows the relative strengths of the GN and LP relaxations, plus relevant problem dimensions for a few of the test problems. The column labeled TIME OF LP/GN displays the time required to obtain the first solution of the LP/GN relaxations, given in IBM 3033 CPU seconds. A special version of GENNET [Ref. 59] was used to obtain the GN relaxations, and the current version of the X System [Ref. 62] was used for the LP relaxations. The GNIP-1 relaxations for a few problems were obtained with the X System so comparative times will not be given. %OPT is the value of the continuous relaxation multiplied by 100 and then divided by the value of the optimal integer solution. The closer %OPT is to 100, the better the relaxation.

STEINER1A and STEINER2A are two SCP's created by transposing the coefficient matrices of STEINER1 and STEINER2 problems. These problems were constructed because no real problems were available having fewer than four non-zero elements per column. The GN relaxations have been reported to be non-competitive with the LP relaxations when the number of non-zeros per column exceeds three or four [Refs. 61]. The results from this study indicate that the strength of the GN relaxation is unpredictable. The GN relaxations for the symmetrical STEINER problems are very competitive with the LP relaxations. The GN relaxation for BUS, however, was unexpectedly bad. The gap for BUS is so great that there is little hope of a reasonable solution trajectory in the enumeration phase. Even the relatively easy problems, TIGER1 and TIGER2, produced GN relaxations of poor quality.

TABLE 5. GENERALIZED NETWORK RELAXATIONS FOR SELECTED PROBLEMS

	MODEL	ROWS/ NODES	COLS/ ARCS	AVG NCE	TIME OF LP/GN	%OPT
STEINER1A	SCP	27	117	3.0	0.28	100.0
	GNIP-2	144	351		0.09	100.0
STEINER2A	SCP	45	330	3.0	1.14	100.0
	GNIP-2	375	990		0.19	100.0
STEINER1	SCP	117	27	13.0	0.09	50.0
	GNIP-2	144	351		0.36	49.6
STEINER2	SCP	330	45	22.0	0.58	50.0
	GNIP-2	375	990		1.44	49.1
BUS	SPP	56	530		8.18	82.8
	GNIP-1	587	3894	6.3	--	0.0014
	GNIP-2	587	3372		1.18	0.0023
TIGER1	SPP	160	636		0.94	100.0
	GNIP-1	797	4769	6.7	--	62.7
	GNIP-2	797	4133		2.11	72.9
TIGER2	SPP	107	2188	4.3	9.46	100.0
	GNIP-2	2296	8292	2.0	3.90	46.3

D. THE GENERALIZED PROCESS NETWORK REFORMULATION

Some recent work by Koene [Ref. 63] on Processing Networks offers an interesting perspective for improving the GNIP formulation. A processing network is one for which the flows on arcs going out from (or into) a given node are proportional to each other. To achieve this proportionality (in our case, we desire equal flows), a network with side constraints must be solved. Several authors have reported some success with solving

pure and generalized networks with a "few" complicating constraints [Refs. 64, 65, 66]. Unfortunately, the number of side constraints in this case grows again with the number of non-zero elements in each column of the ILP. In fact, so many side constraints are needed in these problems that the size of the basis inverse representation which must be carried along with the network becomes prohibitive.

Because the side constraint portion is so large, it was decided to view the generalized process network problem as a candidate for either generalized upper bounding factorization or network factorization routines embedded within an LP system. Generalized Upper Bounding (GUB) refers to a set of rows with at most one non-zero in each column. The coefficients of each non-zero must be +1 or -1 (or capable of being scaled to +1, or -1). Network factorization refers to a set of rows with at most two non-zeros in each column, and the non-zeros may be of any value or sign. Brown and Wright [Ref. 67] have examined techniques for extracting network structures embedded in general LP problems, but the test bed for network factorization routines coupled with an Integer Programming System is not yet in place, so no results can be reported at this time. A Generalized Upper Bounding factorization routine was available, however, and a formulation of the GNIP-2 process network scaled for GUB appears below.

$$(14) \quad \underset{k}{\text{MIN}} \; C_k X_k$$

$$(15) \quad \text{S.T.} \sum_{k:\text{head } j} X_k + \sum_{k:\text{head } j} -Y_k = 0, \; j = 1, \ldots, n; \; \text{(GUB)}$$

$$(16) \quad \sum_{k:\text{tail } i} X_k + \sum_{k:\text{tail } i} \hat{M}_k Y_k = 0, \quad i = 1, \ldots, m;$$

$$(17) \qquad Y_k \;-\; Y_{k'} \;= 0, \qquad i = 1, \ldots, n;\ k \neq k';$$

$$k\text{:tail } i \qquad k'\text{:tail } i$$

where $\hat{M}_{k\text{:head } j} = 1/M_{k\text{:head } j}$.

(14) is the same as for the GNIP-2. (15) forms a GUB set, the right-hand side of (16) would be replaced by $\geq b_i$ for the SCP, and (17) is the side constraint section.

The number of GUB rows attainable is equal to the number of variable nodes in the GNIP. This is an enormous GUB set, but even with the GUB rows not considered, the problem is still larger than the original ILP. It was predicted that the basis inverse representation obtained from this factorization would not be as ill-conditioned as the representation obtained from the normal LP bases. However, the LP's with GUB factorization are also very difficult to solve and the results do not indicate that this promises to be a competitive approach.

VI. COMPUTATIONAL EXPERIENCE

A. THE COMPUTER CODE

The computer code used for this research is a large-scale optimization test bed called the X System or simply XS. XS has been developed since 1974 [Ref. 62] as a general-purpose optimization system of advanced design which serves both as a prototype test bed for research and as the fundamental computational foundation of many application packages utilizing optimization. XS is designed to solve large-scale optimization problems, with special emphasis on mixed integer models. The embedded linear programming module has received most of the design effort and exhibits many singular features including:

1. Hyper-sparse data representation [Ref. 68];
2. Complete constructive degeneracy resolution [Ref. 57];
3. Basis factorization [Ref. 69]; and
4. Elastic range constraints [Ref. 62].

XS consists completely of open FORTRAN subroutines. FORTRAN IV H (Extended) OPTIMIZE (2) is the implementation dialect and an IBM 3033 is the host computer. All of the problems with the exception of AMERICAN were solved interactively under the IBM CMS timesharing system.

1. Elastic Model

XS requires that the model be thought of in an extended or "elastic" sense. The term elastic comes from the view that no constraint is totally binding, but may be violated at a price, an elastic penalty. The feasible region is thereby "stretched" to the degree of elasticity

specified by the penalty structure. The extended elastic model appears below.

$$\text{Minimize} \quad \sum_{j=1}^{n} C_j X_j + \sum_{i=1}^{m} (P_i^- S_i^- + P_i^+ S_i^+)$$

$$\text{s.t.} \quad R_i^- - S_i^- \leq \sum_{j=1}^{n} \delta_{ij} X_j \leq R_i^+ + S_i^+, \qquad i = 1, \ldots, mg;$$

$$R_i^- - S_i^- \leq \sum_{j=1}^{n} a_{ij} X_j \leq R_i^+ + S_i^+, \qquad i = mg + 1, \ldots, m;$$

$$L_j \leq X_j \leq U_j \qquad j = 1, \ldots, n;$$

$$S_i^- \geq 0, \; S_i^+ \geq 0, \qquad i = 1, \ldots, m;$$

where C_j : Cost Coefficients,

a_{ij} : Constraint Coefficients,

P_i^- and P_i^+ : Lower and Upper Constraint Violation Penalties,

R_i^- and R_i^+ : Lower and Upper Constraint Range Limits,

S_i^- and S_i^+ : Logical "artificial" and "surplus" variables,

X_j : Variables (any of which may be integer),

L_j and U_j : Lower and Upper Variable Bounds,

δ_{ij} : Is 0, or +1, or -1 (GUB indicators),

mg : Number of GUB rows,

m : Row Dimension,

n : Column Dimension.

2. Hypersparse Data Representation

Appendix B exhibits a specimen of the data input format used for this research. This particular form exploits the hypersparse data structure capability of XS since there are very few unique real numbers in a SPP/SCP. All of the constraint coefficients are 1 and 0; therefore,

it is not necessary to store a real number for each non-zero coefficient, but merely its address. Further efficiencies can be realized in a similar manner for the unit-cost objective function, and for the right-hand side for which $b_i = 1$ for all i.

3. Primal Dual Algorithm and Degeneracy Resolution

The representation of the Basis Inverse maintained by XS admits the application of the Primal or Dual Algorithm with equal facility. This fact makes it possible to use the Dual Algorithm for this problem class with no loss of performance with respect to the Primal. As mentioned earlier, the Dual is the algorithm of choice because of the massive primal degeneracy present in this class of problems.

It is this equal facility between the Primal and the Dual which provides the framework for the degeneracy resolution machinery in XS. In Graves' terminology [Ref. 57], when degeneracy is encountered, the algorithm is said to be "blocked." The resolution of blocking in either the primal or dual algorithm is accomplished by shifting to the alternate algorithm when blocking occurs. The alternate algorithm is applied to a subproblem of the original problem and at worst we are led to a contracting sequence of problems to which we alternately apply the primal and dual algorithms. A strict contraction can be assured, and thus in at most a finite number of steps, resolution is assured. A complete illustration of blocking resolution can be found in [Ref. 70].

The degree to which blocking is resolved through this sequential nesting of subproblems is controlled by a blocking resolution parameter. This parameter can be set so that any degree between no resolution and total resolution can be obtained. The parameter also controls the point

at which blocking resolution begins in the solution trajectory. This means that blocking resolution can be inhibited early in the solution trajectory when it may not be efficient to resolve every degeneracy, and then enabled as the trajectory nears optimality.

B. HEURISTICS WITHIN THE EXACT ALGORITHM

The efficiency of branch and bound algorithms can be improved through judicious use of heuristic information that indicates good branching strategies to follow. If good feasible solutions (incumbents) can be obtained early, then fathoming can occur more quickly and more frequently in the search. Also, premature termination of the algorithm will more often result in near-optimal or optimal solutions.

1. The Elastic Heuristic

A robust technique for obtaining an incumbent has been incorporated into the XS enumeration system. Any continuous relaxation of the Elastic ILP can be rounded to an integer solution with very little computational effort. Further, all such rounded solutions are admissable (feasible in the extended elastic sense). The current continuous solution is rounded in three passes, each of which selects variables from a class defined in terms of θ, where $0 \leq \theta \leq .5$ and $(1 - \theta) \leq X_j \leq 1$ or $0 \leq X_j \leq \theta$.

Class 1: Nearly integral $(0 \leq \theta \leq .2)$.

Class 2: Fractional $(.2 \leq \theta \leq .4)$.

Class 3: Ambivalent $(.4 \leq \theta \leq .5)$.

The rounding heuristic sequentially exhausts variables from each class and rounds using a "minimal regret function," rounding away from the worst penalty.

In the default elastic enumeration scheme, there are only depth and value-motivated fathoming rules. Feasibility plays no direct role in the enumeration except via the accumulated cost of the penalties in the objective function. Integer solutions and lower bounds of excellent quality are empirically produced quite early in the enumeration effort, permitting routine early termination of the search based on an optimality tolerance or on a maximum depth (permissible number of fixed variables in any restriction). Tuning of the method is easily accomplished via these two limits and the elastic penalties used to express the underlying model.

The penalty structure found most effective with this heuristic is based upon the number of non-zero row elements in each row of the constraint matrix, called NRE(i). It has also been determined that the upper penalty P^+ should be set at one-half the value of the lower penalty P^-, in order to coerce the heuristic to round up. This forces shallow termination with respect to depth fathoming. A penalty constant, P, is set at approximately one order of magnitude greater than the largest cost coefficient, so that for each row in the SPP/SCP

Set $P_i^- = P/NRE(i)$.

Set $P_i^+ = \frac{1}{2} P_i^-$.

Penalties set in this manner "communicate" to the heuristic that rows which can be covered in only a few ways are more important than rows with a higher row count. In the enumeration, then, these "important" rows are satisfied first, making it possible to avoid many of the alternate possibilities available for covering rows with a large row count.

Table 6 exhibits the computational characteristics of the Elastic Method. LP and LPtime indicate the solution value and solution time (in

CPU seconds) for the first continuous relaxation of the indicated problems. OPT is the optimal integer solution, and ILPtime is the total CPU time in seconds required to solve the ILP to the optimum. INPUT/OUTPUT time is included in the ILPtime values. Input time includes the time to read in the entire problem and accomplish error checking. Output time includes the printing of intermediate information plus the time required to execute the report writer. The dual algorithm was used for all solutions.

TABLE 6. COMPUTATIONAL RESULTS FOR THE ELASTIC METHOD

	LP	LPtime	OPT	ILPtime
STEINER1	9.0	0.08	18	25.13
MCOVER1	0.0 (9)	0.05	0 (18)	0.39
STEINER2	15.0	0.52	30	527.08
MCOVER2	0.0 (15)	0.22	0 (30)	417.08
TIGER1	56406.0	0.94	56406	0.97
TIGER2	15098.0	9.46	15098	9.53
TIGER2a	15684.0	10.23	15684	10.30
BUS	50754.5	8.18	61308	177.44
AMERICAN	No LP solution after 30 minutes CPU time			
TRUCK	No LP solution after 30 minutes CPU time			
TANKER	75941.4	34.05	75941.4	44.62

It is interesting to note that the maximal set covering reformulations of the two STEINER problems as MCOVER1 and MCOVER2 produced easier ILP'S. The budget constraints for these problems were constructed so that the maximal covering problems would seek the same optima as the SCP formulations. TIGER2a is a variant of TIGER2 obtained by eliminating

some of the columns found in the optimal solution of TIGER2. Marsten has indicated in [Ref. 23] that problems in this general class typically have many nearly optimal integer solutions. TIGER2a supports this observation: 8 of the 33 optimal columns for TIGER2 were removed to construct TIGER2a, and the solution is only 3.9 percent worse.

C. THE ELASTIC METHOD WITH STARTING SOLUTIONS

As indicated in Table 6, the LP relaxations for AMERICAN and TRUCK were not solved within 30 minutes. After many unsuccessful attempts to overcome the numerical instabilities peculiar to these LP's, various combinatorial and heuristic methods for obtaining starting solutions were considered. Given a feasible, suboptimal solution of "reasonable" quality, it should be possible for the elastic method to find the optimal solution in few enough iterations to avoid many of the numerical difficulties.

D. LOGICAL REDUCTION

Garfinkel and Nemhauser [Ref. 71] have given a set of simple rules for logically reducing a problem matrix for the SPP/SCP with $b_i = 1$ for all i. Although logical reduction is not guaranteed to provide a starting solution, substantial reductions in problem size can greatly improve the numerical behavior of many problems, especially if the problems were originally generated with many inherent redundancies. This explanation of logical reduction provides insight into the structure of the SPP/SCP and is valuable for understanding and evaluating some of the heuristics used for constructing starting solutions.

Not all of the rules were chosen for implementation because it is felt that their inclusion does not return sufficient reduction to justify

the computational expense of using them. The implementation scheme used is a non-backtracking (polynomial time) routine involving easy binary comparisons of rows and columns. If significant reduction is achieved after one application, the scheme is applied iteratively until no more improvement is obtained. Those rules which were implemented are based on the notions of row and column dominance. For example, if two columns are equal element-by-element (ColA = ColB), and one has a cheaper cost, then the more expensive column is dominated by the cheaper and may be deleted. Not so obviously, if RowA is wholly contained as a subset of RowB (RowA $\leq$ RowB), then RowB may be deleted, since any column which covers RowA will also cover RowB. The four rules used are:

Rule 1: Delete all null columns and null rows.

Rule 2: Column Dominance.

- A. SCP: If ColA $\geq$ ColB, and CostA $\leq$ CostB, delete ColB.
- B. SPP: If ColA = ColB, delete the more expensive column.

Rule 3: Row Singleton.

- A. Delete the row covered by only one column.
- B. Fix the variable associated with the singleton to one.
- C. Delete all rows covered by the fixed variable.
- D. SPP: Delete all columns in the rows deleted by 3C.

Rule 4: Row Dominance.

- A. If RowA $\geq$ RowB, delete RowA.
- B. SPP: If RowA is deleted, also delete every column in RowA which is not included in RowB.

The column and row reduction schemes can be applied iteratively, since after the first application, additional dominance may be discovered.

Consider the constraint matrix of the Air Freight Example below.

Application of the reduction rules would achieve the following results:

	R1	R2	R3	R4	R5	R6	R7
Los Angeles	1	0	0	0	0	0	0
San Francisco	1	1	1	0	0	0	0
San Jose	1	1	1	0	0	0	0
Denver	0	0	1	1	1	0	0
Portland	0	0	0	1	1	0	0
Seattle	0	0	0	1	1	1	0
San Diego	0	0	0	0	1	1	1
Costs	0	0	0	0	6	7	4

For the SPP,

Rule 3: A. Delete Los Angeles.

B. Fix R1 to one and delete it.

C. Delete San Francisco and San Jose.

D. Delete R2 and R3.

Rule 4: A. Denver > Portland, delete Denver

Seattle > Portland, delete Seattle

B. Delete R6.

The resulting reduced problem is

	R4	R5	R7
Portland	1	1	0
San Diego	0	1	1
Costs	0	6	4

Solution = R1, R4, R7.

The action of deleting a row or column does not necessarily mean that the row or column has been eliminated from the problem. In the reduced air freight example, the removal of Denver and Seattle from consideration means simply that any column which covers Portland will automatically cover Denver and Seattle; therefore, Portland is the critical row. The same observation holds for San Diego. Table 7 exhibits the degree of reduction achieved and the computation times for two of the test problems. % RED is derived by dividing the number of rows/columns deleted by the number of original rows/columns, respectively. No reduction was achieved for STEINER1, STEINER2, BUS, TIGER2, TANKER, and TRUCK. TIME indicates the total time in CPU seconds required to achieve the indicated reduction.

TABLE 7. LOGICAL REDUCTION RESULTS FOR SELECTED PROBLEMS

	ROWS	% RED	COLS	% RED	ITERATIONS	TIME
TIGER1	160	50	636	13.8	1	27.3
AMERICAN	95	0	9318	47.0	1	1743.0

The tremendous column reduction achieved on AMERICAN is due to the absence of crew base constraints. During the original generation of this problem, entire sets of columns were replicated and were designed to be kept distinct by the mutually exclusive crew base constraints. Unfortunately, the original crew base constraints are no longer available. Once the reduction for AMERICAN is explained, then, the benefits obtainable by logical reduction do not appear to justify its computational expense.

The reduction routine could be a valuable aid, however, in validating the performance of column generation programs, and could also be of use in the first screening of problem data.

E. A GREEDY HEURISTIC ALGORITHM

Baker [Ref. 72] describes a heuristic algorithm developed to exploit the structure of large airline crew scheduling problems formulated as SCP's. The approach is to successively augment the solution set for the SCP by selecting columns which exhibit the minimum average cost per uncovered row.

STEP 1: Initialization. Solution set = $\emptyset$. Row Coverage Set = $\emptyset$.

STEP 2: Selection. Choose the column X_j^* that has the minimum average cost per uncovered row.

STEP 3: Update. Add X_j^* to the solution set. Update the row coverage set to reflect the rows covered by X_j^*. If all rows are covered, STOP. Otherwise GO TO STEP 2.

The worst case bound for the solution obtained from this procedure is reported by Baker to be:

$$\text{SOLN(Heuristic)} \leq \text{SOLN(Opt)} \sum_{k=1}^{E} 1/k \ ,$$

where E is the maximum number of non-zero elements in any column in the solution set. This means that for a set of columns with from four to ten non-zero elements, the worst solution obtainable from the heuristic is from two to three times larger than the value of the optimum. Table 8 indicates, however, that the actual performance of the heuristic can be much better than the worst case bound. The column labeled START TIME indicates the time required to obtain the starting solution. %OPT is the percentage difference between the starting solution and the optimal solution. This simple heuristic will provide a classically feasible solution of reasonable quality for the SCP. The solution for SPP

will not be feasible, since the heuristic will treat the SPP as a SCP and overcovering of the rows will result.

TABLE 8. STARTING SOLUTIONS OBTAINED FROM THE BAKER SCP HEURISTIC

	START TIME	% OPT
STEINER1	0.07	5.6
STEINER2	0.06	6.7
TRUCK	8.47	9.19

Because these starting solutions are of limited value for set partitioning problems, this line of study was terminated in favor of a new solution technique developed by Brown and Graves [Ref. 73]. The new technique uses a block partitioning scheme to exploit the intrinsic structure of the SPP/SCP.

F. BLOCK PARTITIONING ALGORITHM

Christofides [Ref. 3] describes a block partitioning structure attributed to Pierce [Ref. 15] which has been used by many researchers for this problem class. To place the SPP/SCP in block form, we make up m blocks of columns, one block for each row. Block i will comprise of exactly those columns which cover row i, but do not cover rows 1 to i-1. This produces a staircase matrix with zeros to the right of the staircase. The blocks in general can be arranged in tableau form as shown in Figure 3, although one or more blocks may be nonexistent in a particular problem.

Marsten [Ref. 1] determined experimentally that sorting the rows by increasing length gave consistently good results for his algorithm which favors the shorter rows for early branching. The row with the fewest

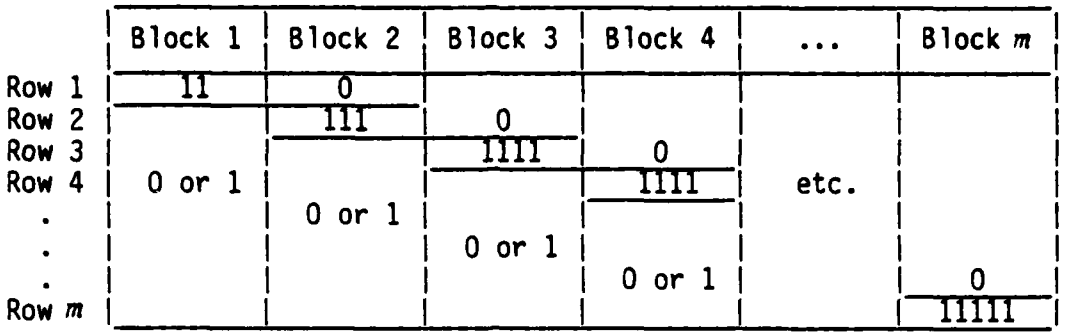


Figure 3. Block Partition Structure

1's is placed at the top, and the row with the most 1's ends up at the bottom. (This ordering by row length is also depicted in Figure 3.) Intuitively, it seems reasonable that a row which can be covered in only a few ways is more critical than a row which can be covered in many ways, and should therefore be dealt with first. This row ordering scheme was chosen for implementation.

Once the problem has been placed into the block structure, three ways of ordering columns within blocks are found in the literature: (1) heuristically by increasing or decreasing cost [Ref. 3]; (2) lexicographically [Ref. 1]; and (3) randomly (i.e., columns are not explicitly reordered once blocking has been accomplished). The algorithm developed by Brown and Graves does not presently require that columns be specially ordered within blocks.

The Block Partitioning Algorithm can be applied to both the initial LP Relaxation and subsequently to the integer enumeration. For the LP, the problem is first divided into an arbitrary number of block groups forming a set of distinct LP subproblems. The first LP subproblem (LP_1)

is solved to optimality, the next LP subproblem (LP_2) is dynamically appended to the first, and then the two subproblems are solved as one. The third LP subproblem is appended to the solution of $LP_{1,2}$ and the procedure continues until all subproblems have been appended and a global solution has been obtained.

Many variations of this procedure are evident. TRUCK has been solved by dual relaxation of the aggregation of successive LP solutions. Particular problems exhibit great sensitivity to tuning of this procedure. In particular, a few complicating columns are frequently the principal cause of computational difficulty.

Table 9 presents results for the Block Partitioning Algorithm applied to the LP only. Subsequent to the solution of the global LP, the elastic enumeration scheme was used to obtain optimality. OPT TIME indicates the total time required to achieve optimality. I/O time is included in the values. All times are in IBM 3033 CPU seconds.

TABLE 9. RESULTS FOR THE BLOCK PARTITIONING ALGORITHM

	NUMBER OF BLOCKS	BLOCK TIME	NUMBER OF LP SUBPROBLEMS	GLOBAL LP Time	OPT TIME
BUS	48	0.001	4	10.76	174.50
TIGER2	106	0.04	4	10.98	15.98
TRUCK	194	0.15	4	210.92*	373.72*
TANKER	50	0.11	4	9.14	22.56
AMERICAN	79	0.24	4	132.52	536.41

*Primal Feasible, Suboptimal solutions.

The analogous procedure whereby each subproblem is solved to an integer optimum did not compare favorably with the default elastic method. This line of study was therefore terminated and another concept called the "Refinement Procedure" was investigated.

G. COLUMN GENERATION AND PROBLEM REFINEMENT

There is no substitute for possessing the column generating program when attempting to solve the large-scale SPP/SCP. Attempting to solve large, static problems in a vacuum is doomed to be either expensive or impossible. The column generator and the optimization system work best on these problems when they are intimately coupled so that each module can communicate with the other. In this way, the optimizer works on smaller problems and the column generator produces only those columns which can contribute to a better solution.

Graves [Ref. 73] has developed a refinement procedure for the SPP which attempts to capture, for a static problem, some of the capabilities which are present when the column generator is in hand. This procedure results in a relaxation of the original problem, but it is a workable scheme which can produce acceptable solutions. The procedure is implemented as follows:

STEP 1: Solve the SPP as a SCP. Identify rows with multiple covers. If no multiple covers exist, STOP.

STEP 2: For each column covering a row which is multiply covered, generate a new column which does not include rows with multiple covers. Original columns are assigned a "cost per row covered" which is used to give new columns reduced costs proportionate to the number of rows deleted. Go to STEP 1.

Table 10 displays the performance of the refinement procedure on two of the more difficult SPP's, BUS and AMERICAN. The refinement procedure is

used in conjunction with the Block Partitioning Algorithm set for four LP subproblems. # REFINEMENTS gives the number of refinement iterations, # COLUMNS GENERATED gives the total number of new columns generated by the procedure, and OPT TIME is the total time in CPU seconds required to achieve the optimal partition.

TABLE 10. RESULTS FOR THE REFINEMENT PROCEDURE

	# REFINEMENTS	# COLUMNS GENERATED	OPT TIME
BUS	5	123	12.06
AMERICAN	2	26	94.60

Comparing the results from Tables 9 and 10, it is obvious that the refinement procedure produces a solution much more quickly than the other methods. It is difficult to compare the solution values, because the true cost for each column generated by the procedure is not known. The costs assigned here to the new columns are representative, however, of those which would be assigned by the column generator.

VIII. CONCLUSIONS AND RECOMMENDATIONS

It has been shown that practical, large-scale set covering and set partitioning problems can be solved optimally and efficiently. The Block Partitioning Algorithm is clearly the most robust and most successful technique examined in this study, and its efficiency compares favorably with published solution technologies for this problem class.

Unfortunately, the implementation of the Generalized Network Reformulation for the SPP/SCP did not perform as well as expected. The continuous relaxation of the Integer Generalized Network is too weak to be of much practical use; therefore, this technique does not hold much promise for the rapid solution of set covering and set partitioning problems.

Much work remains in improving the integer enumeration scheme subsequent to the solution of the linear programming relaxation. The default elastic method works well, but additional research is needed to improve its performance. The coupling of the column generating program with the optimizer is a concept which holds great promise for the efficient solution of problems in this class. As illustrated by the Refinement Procedure results, spectacular reductions in solution time can result from implementing this idea.

The proposed standard data input format displayed in Appendix B makes data manipulation both easy and convenient, and dramatically reduces storage requirements for any mathematical programming system capable of exploiting it. A tape containing all of the test problems in this format is available to those who wish to continue research in this area.

APPENDIX A. DESCRIPTION OF SELECTED APPLICATIONS

A. POLITICAL DISTRICTING (SPP)

Let the rows represent m basic population units (such as counties, census tracts, etc.). Let the columns represent n possible districts or subsets of the population units such that each potential district meets the requirements on population size, contiguity, compactness, and so forth. A side cardinality condition (9) usually imposed is that there be exactly J districts. If C_j is some ordinal measure of the unacceptability of district j, then an optimal solution to the SPP yields an optimal districting plan.

B. COLORING PROBLEMS (SPP)

Consider the problem of coloring a map so that no two adjacent areas have the same color. Let there be m such areas. A column j is generated if no two elements of column j correspond to areas having a common boundary. If all costs are unity, an optimal partition indicates the minimum number of colors needed. A direct application of this concept is the problem of minimizing the number of distinct radio frequencies necessary to provide service in several geographical areas. A column j is generated if no two elements of column j correspond to areas with overlapping frequencies.

C. NUCLEAR AND CONVENTIONAL TARGETING

1. Conventional Scenario (SCP)

Let each row i represent a target which must be engaged at least b_i times. Let each column j represent a weapons system capable of engaging some subset of the m targets within a specified time period. If the cost coefficients reflect the expected effectiveness of a given weapons system on the targets covered by column j, the optimal solution will yield the most effective subset of weapons systems capable of accomplishing the mission. If columns are generated so that k missions are possible for each of p weapons systems, then a constraint will be necessary to ensure that each weapon system is given only one mission in the optimal solution. The maximal SCP formulation can also be used here to find the combination of weapons systems which can engage some specified proportion of targets.

2. Nuclear Scenario (SPP)

Let each row represent a target which must be engaged only once in a given time period (to avoid fratricide, for instance). Let each column j represent a weapons system capable of engaging some subset of the m targets (i.e., various footprint alternatives). If a unit-cost objective function is used, the optimal solution will yield the minimum number of weapons systems needed to destroy all the targets.

D. INFORMATION RETRIEVAL (SCP)

Consider the problem of retrieving information from n files, where the j^{th} file is of length C_j. Suppose that m requests for information are received. Each unit of information is stored in at least one file j indicated by $a_{ij} = 1$. An optimal cover yields a subset of files that

minimizes the maximum total length which needs to be searched in order to guarantee retrieval of all the information.

E. CYCLIC SCHEDULING (SCP)

A fundamental problem of cyclic staffing is to size and schedule a minimum cost workforce so that sufficient workers are on duty during each time period. The k,m cyclic scheduling problem models the task of finding the minimum cost assignment of workers to shifts so that each person works k time periods consecutively out of m, and at least b_i workers are present during the day i. A sample tableau for the 5,7 cyclic scheduling problems is shown below.

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	RHS
MON	1	0	0	1	1	1	1	$\geq b_1$
TUE	1	1	0	0	1	1	1	$\geq b_2$
WED	1	1	1	0	0	1	1	$\geq b_3$
THU	1	1	1	1	0	0	1	$\geq b_4$
FRI	1	1	1	1	1	0	0	$\geq b_5$
SAT	0	1	1	1	1	1	0	$\geq b_6$
SUN	0	0	1	1	1	1	1	$\geq b_7$

$$0 \leq X_j \leq U_j \text{ and Integer for all } j$$

The above formulation is not a binary program; therefore, to transform it into one, two alternative techniques can be used. If it is desired to distinguish between individual workers, the above seven columns can be replicated for each worker. An additional side constraint will be necessary to ensure that a worker is not selected to work more than one

shift. This approach could result in a large number of columns, so another alternative is to use a binary representation in place of each X_j. For any reasonable value of U_j, this is a feasible technique. For large values of U_j, the requirement that the X_j be integer is probably not worth the computational expense, and the problem should be solved as a continuous LP. Additional considerations such as overtime, days-off scheduling, part-time workers, over- and under-staffing, etc., are discussed in [Ref. 51].

F. SALES TERRITORY DESIGN (SPP)

A problem facing sales managers is how to identify which customers should be included in a given sales territory, and how to determine the best call frequencies for individual customers, in short, how to allocate a given amount of the time of several salesmen to several hundred prospective customers so as to maximize sales. Let the rows be customers. Let the columns represent p sets of candidate territories, one for each of the p salesmen. Let the costs reflect the potential sales response evaluations for a particular salesman in territory j. A side constraint is necessary to ensure that only one territory is picked from each of the p sets. The requirement that a customer can appear in one and only one sales territory makes this the SPP.

The generation of candidate territories is a difficult process in itself. Shanker, et al. [Ref. 8] suggest a procedure which involves solving a series of integer programs. One ILP selects territories which maximize demand potential subject to a series of workload, stratification, and compactness constraints. This set of territories is in turn evaluated

in another ILP which maximizes a piecewise linear response function subject to calling frequency constraints. Subjective considerations can be included at various points in the process to help further reduce the number of candidate territories which finally appear in the SPP.

APPENDIX B. PROPOSED DATA INPUT FORMAT

A very large part of the time invested in this research has been spent manipulating and entering problem data which came to the author in many different forms. The sheer size of the data sets made them extremely unwieldy; therefore, it was decided early on that a compact format for these problems could make data manipulation both easy and convenient, and would encourage other researchers to adopt this format as a standard. The format chosen has many advantages for large-scale problems.

1. It is compact, listing only problem dimensions, constraint ranges, cost coefficients, and coefficient addresses. This not only reduces Input/Output time, but makes it possible to handle quite large data sets under interactive, time-sharing systems such as IBM CMS.

2. Storage requirements are easily calculated. Problem dimensions are known immediately after reading the first card image. This eliminates the need to make multiple passes of the data, or to guess at the problem size, as is the case with MPS format [Ref. 74].

3. Data Generation Programs are simplified. Row and column labels are accommodated, but they are not primary keys, thus avoiding alpha-numeric manipulations with symbol tables.

4. Column manipulation of data input is made easy since all information for each column is contiguous.

5. This column format is easily generated by commercially available (MPS) problem generation systems.

The data input format consists of three sets of card images:

1. Problem Dimensions. Format (3I6) (One Card)

   a. M = Number of Rows

   b. N = Number of Columns

   c. NZEL = Number of non-zero Elements.

2. Constraint Ranges. Format (2A4, 2E16.8) (M Cards)

   a. IR = Row Index

   b. RL = Lower Range Limit

   c. RU = Upper Range Limit.

3. Column Data. (N or More Cards)

   a. The number of cards needed depends upon the number of non-zero elements in each column (= NCE). The format for the first column card is (2A4, F14.3, 10I5).

      1. JC = Column Index

      2. C_j = Column Cost Coefficient

      3. NCE = Number of Non-Zero elements in the column

      4. IR = Row Addresses of Non-zero Coefficients.

   b. If NCE is greater than 9, additional column cards are needed to hold the row addresses for that column. The format for additional column cards is (20x, 10I5).

TABLE 11. INPUT DATA FOR STEINER1

AN EXAMPLE IN PROPOSED STANDARD FORMAT

```
117     27   351
  1  0.100000000D+01   0.100000000D+21
  2  0.100000000D+01   0.100000000D+21
  3  0.100000000D+01   0.100000000D+21
  4  0.100000000D+01   0.100000000D+21
  5  0.100000000D+01   0.100000000D+21
  6  0.100000000D+01   0.100000000D+21
  7  0.100000000D+01   0.100000000D+21
  8  0.100000000D+01   0.100000000D+21
  9  0.100000000D+01   0.100000000D+21
 10  0.100000000D+01   0.100000000D+21
 11  0.100000000D+01   0.100000000D+21
 12  0.100000000D+01   0.100000000D+21
 13  0.100000000D+01   0.100000000D+21
 14  0.100000000D+01   0.100000000D+21
 15  0.100000000D+01   0.100000000D+21
 16  0.100000000D+01   0.100000000D+21
 17  0.100000000D+01   0.100000000D+21
 18  0.100000000D+01   0.100000000D+21
 19  0.100000000D+01   0.100000000D+21
 20  0.100000000D+01   0.100000000D+21
 21  0.100000000D+01   0.100000000D+21
 22  0.100000000D+01   0.100000000D+21
 23  0.100000000D+01   0.100000000D+21
 24  0.100000000D+01   0.100000000D+21
 25  0.100000000D+01   0.100000000D+21
 26  0.100000000D+01   0.100000000D+21
 27  0.100000000D+01   0.100000000D+21
 28  0.100000000D+01   0.100000000D+21
 29  0.100000000D+01   0.100000000D+21
 30  0.100000000D+01   0.100000000D+21
 31  0.100000000D+01   0.100000000D+21
 32  0.100000000D+01   0.100000000D+21
 33  0.100000000D+01   0.100000000D+21
 34  0.100000000D+01   0.100000000D+21
 35  0.100000000D+01   0.100000000D+21
 36  0.100000000D+01   0.100000000D+21
 37  0.100000000D+01   0.100000000D+21
 38  0.100000000D+01   0.100000000D+21
 39  0.100000000D+01   0.100000000D+21
 40  0.100000000D+01   0.100000000D+21
 41  0.100000000D+01   0.100000000D+21
 42  0.100000000D+01   0.100000000D+21
 43  0.100000000D+01   0.100000000D+21
 44  0.100000000D+01   0.100000000D+21
 45  0.100000000D+01   0.100000000D+21
 46  0.100000000D+01   0.100000000D+21
 47  0.100000000D+01   0.100000000D+21
 48  0.100000000D+01   0.100000000D+21
 49  0.100000000D+01   0.100000000D+21
 50  0.100000000D+01   0.100000000D+21
 51  0.100000000D+01   0.100000000D+21
 52  0.100000000D+01   0.100000000D+21
 53  0.100000000D+01   0.100000000D+21
 54  0.100000000D+01   0.100000000D+21
 55  0.100000000D+01   0.100000000D+21
 56  0.100000000D+01   0.100000000D+21
 57  0.100000000D+01   0.100000000D+21
 58  0.100000000D+01   0.100000000D+21
 59  0.100000000D+01   0.100000000D+21
 60  0.100000000D+01   0.100000000D+21
```

61	0.100000000D+01	0.100000000D+21
62	0.100000000D+01	0.100000000D+21
63	0.100000000D+01	0.100000000D+21
64	0.100000000D+01	0.100000000D+21
65	0.100000000D+01	0.100000000D+21
66	0.100000000D+01	0.100000000D+21
67	0.100000000D+01	0.100000000D+21
68	0.100000000D+01	0.100000000D+21
69	0.100000000D+01	0.100000000D+21
70	0.100000000D+01	0.100000000D+21
71	0.100000000D+01	0.100000000D+21
72	0.100000000D+01	0.100000000D+21
73	0.100000000D+01	0.100000000D+21
74	0.100000000D+01	0.100000000D+21
75	0.100000000D+01	0.100000000D+21
76	0.100000000D+01	0.100000000D+21
77	0.100000000D+01	0.100000000D+21
78	0.100000000D+01	0.100000000D+21
79	0.100000000D+01	0.100000000D+21
80	0.100000000D+01	0.100000000D+21
81	0.100000000D+01	0.100000000D+21
82	0.100000000D+01	0.100000000D+21
83	0.100000000D+01	0.100000000D+21
84	0.100000000D+01	0.100000000D+21
85	0.100000000D+01	0.100000000D+21
86	0.100000000D+01	0.100000000D+21
87	0.100000000D+01	0.100000000D+21
88	0.100000000D+01	0.100000000D+21
89	0.100000000D+01	0.100000000D+21
90	0.100000000D+01	0.100000000D+21
91	0.100000000D+01	0.100000000D+21
92	0.100000000D+01	0.100000000D+21
93	0.100000000D+01	0.100000000D+21
94	0.100000000D+01	0.100000000D+21
95	0.100000000D+01	0.100000000D+21
96	0.100000000D+01	0.100000000D+21
97	0.100000000D+01	0.100000000D+21
98	0.100000000D+01	0.100000000D+21
99	0.100000000D+01	0.100000000D+21
100	0.100000000D+01	0.100000000D+21
101	0.100000000D+01	0.100000000D+21
102	0.100000000D+01	0.100000000D+21
103	0.100000000D+01	0.100000000D+21
104	0.100000000D+01	0.100000000D+21
105	0.100000000D+01	0.100000000D+21
106	0.100000000D+01	0.100000000D+21
107	0.100000000D+01	0.100000000D+21
108	0.100000000D+01	0.100000000D+21
109	0.100000000D+01	0.100000000D+21
110	0.100000000D+01	0.100000000D+21
111	0.100000000D+01	0.100000000D+21
112	0.100000000D+01	0.100000000D+21
113	0.100000000D+01	0.100000000D+21
114	0.100000000D+01	0.100000000D+21
115	0.100000000D+01	0.100000000D+21
116	0.100000000D+01	0.100000000D+21
117	0.100000000D+01	0.100000000D+21

1	1.000	13	2	3	7	10	37	38	39	40	41
		42	43	44	45						
2	1.000	13	1	3	8	11	46	47	48	49	50
		51	52	53	54						
3	1.000	13	1	2	9	12	55	56	57	58	59
		60	61	62	63						
4	1.000	13	1	5	6	10	64	65	66	67	68
		69	70	71	72						
5	1.000	13	2	4	6	11	73	74	75	76	77
		78	79	80	81						
6	1.000	13	3	4	5	12	82	83	84	85	86
		87	88	89	90						
7	1.000	13	4	8	9	10	91	92	93	94	95
		96	97	98	99						
8	1.000	13	5	7	9	11	100	101	102	103	104
		105	106	107	108						
9	1.000	13	6	7	8	12	109	110	111	112	113
		114	115	116	117						
10	1.000	13	14	15	19	22	37	46	55	64	73
		82	91	100	109						
11	1.000	13	13	15	20	23	38	47	56	65	74
		83	92	101	110						
12	1.000	14	13	14	21	24	39	48	57	66	67
		75	84	93	102	111					
13	1.000	12	13	17	18	22	40	49	58	76	85
		94	103	112							
14	1.000	13	14	16	18	23	41	50	59	68	77
		86	95	104	113						
15	1.000	13	15	16	17	24	42	51	60	69	78
		87	96	105	114						
16	1.000	13	16	20	21	22	43	52	61	70	79
		88	97	106	115						
17	1.000	13	17	19	21	23	44	53	62	71	80
		89	98	107	116						
18	1.000	13	18	19	20	24	45	54	63	72	81
		90	99	108	117						
19	1.000	13	26	27	31	34	37	51	59	70	75
		83	94	108	116						
20	1.000	13	25	27	32	35	42	47	58	66	80
		82	99	104	115						
21	1.000	13	25	26	33	36	41	49	57	65	73
		90	98	106	114						
22	1.000	13	25	29	30	34	43	48	56	67	81
		89	91	105	113						
23	1.000	13	26	28	30	35	39	53	55	72	77
		88	96	101	112						
24	1.000	13	27	28	29	36	38	46	63	71	79
		87	95	103	111						
25	1.000	13	28	32	33	34	40	54	62	64	78
		86	97	102	110						
26	1.000	13	29	31	33	35	45	50	61	69	74
		85	93	107	109						
27	1.000	13	30	31	32	36	44	52	60	68	76
		84	92	100	117						

LIST OF REFERENCES

1. Marsten, R. E., "An Algorithm for Large Set Partitioning Problems," Management Science, Vol. 20, pp. 774-787, 1974.

2. Christofides, N., and Korman, S. M., "A Computational Survey of Methods For The Set Covering Problem," Management Science, Vol. 21, No. 5, January 1975.

3. Glover, F., and Mulvey, J. M., "Equivalence of the 0-1 Integer Programming Problem to Discrete Generalized and Pure Networks," Operations Research, Vol. 28, No. 3, 1980.

4. Wagner, H. M., Principles of Operations Research, Prentice-Hall, 1975.

5. Gaver, D. P., and Thompson, G. L., Programming and Probability Models In Operations Research, Brooks/Cole, 1973.

6. Lin, Chien-Hua, "Corporate Tax Structures and a Special Class of Set-Partitioning Problems," Ph.D. Dissertation, Case Western Reserve University, 1975.

7. Marsten, R. E., and Muller, M. R., "A Mixed Integer Programming Approach To Air Cargo Fleet Planning," Management Science, Vol. 26, No. 11, pp. 1096-1107, 1980.

8. Shanker, R. J., Turner, R. E., and Zoltners, A. A., "Sales Territory Design: An Integrated Approach," Management Science, Vol. 22, No. 3, November 1975.

9. Cullen, F. H., Jarvis, J. J., and Ratliff, H. D., "Set Partitioning Based Hueristics For Interactive Routing," Networks, Vol. 11, pp. 125-143, 1981.

10. Geoffrion, A. (ed.), Perspectives on Optimization, Addison-Wesley, 1972.

11. Balas, E., and Padberg, M. W., "Set Partitioning: A Survey," in N. Christofides (ed.), Combinatorial Optimization, Wiley, 1979.

12. Dantzig, G. B., and Ramser, J. H., "The Truck Dispatching Problem," Management Science, Vol. 6, No. 1, pp. 80-91, 1959.

13. Balinski, M. L. and Quandt, R., "On an Integer Program for a Delivery Problem," Operations Research, Vol. 12, pp. 300-304, 1964.

14. Clark, G. and Wright S. W., "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points," Operations Research, Vol. 12, No. 4, pp. 568-581, 1964.

15. Pierce, J. F., "Application of Combinatorial Programming to a Class of All-Zero-One Integer Programming Problems," Management Science, Vol. 15, pp. 191-209, 1968.

16. Laskey, J. S., "Optimal Scheduling of Freight Trucking," M.S. Thesis, Massachusetts Institute of Technology, June 1968.

17. Ronen, D., "Scheduling of Vessels for Shipment of Bulk and Semi-bulk Commodities Originating in a Single Area," Ph.D. Dissertation, Ohio State University, 1979.

18. Spitzer, M., "Solution to the Crew Scheduling Problem," presented at the First AGIFORS Symposium, October 1961.

19. Kolner, T. N., "Some Highlights of a Scheduling Matrix Generation System," United Airlines, presented at the Sixth AGIFORS Symposium, September 1966.

20. Arabeyre, J. P., Fearnley, J., Steiger, F. C., and Teather, W., "The Airline Crew Scheduling Problem: A Survey," Transportation Science, Vol. 3, No. 2, pp. 140-163, 1969.

21. Thiriez, H., "Airline Crew Scheduling: A Group Theoretic Approach," Ph.D. Dissertation, Massachusetts Institute of Technology, October 1969.

22. Marsten, R. E., Muller, M. R., and Killion, C. L., "Crew Planning at Flying Tiger: A Successful Application of Integer Programming," Management Science, Vol. 25, pp. 1175-1183, 1979.

23. Marsten, R. E. and Shephardson, F., "Exact Solution of Crew Scheduling Problems Using the Set Partitioning Model: Recent Successful Applications," Networks, Vol. 11, pp. 165-177, 1981.

24. ReVelle, C., Toregas, C., and Falkson, L., "Applications of the Location Set-covering Problem," Geographical Analysis, Vol. 7, 1976.

25. Gleason, J. M., "Set Covering Approach to the Location of Express Bus Stops," Paper given at ASME Meeting, September, 1973.

26. Revelle, C., Marks, D., and Liebman, J. C., "An Analysis of Private and Public Sector Location Models," Management Science, Vol. 16, No. 12, pp. 692-707, 1970.

27. Plane, D. R. and Hendrick, T. E., "Mathematical Programming and the Location of Fire Companies for the Denver Fire Department, Operations Research, Vol. 25, No. 4, pp. 563-578, 1977.

28. Moore, G. C. and Revelle, C., "The Hierarchical Service Location Problem," Management Science, Vol. 28, No. 7, July 1982.

29. Levin A., "Fleet Routing and Scheduling Problems for Air Transportation Systems," Ph.D. Dissertation, Massachusetts Institute of Technology, January 1969.

30. Dwyer, F. R. and Evans, J. R., "A Branch and Bound Algorithm for the List Selection Problem in Direct Mail Advertising," Management Science, Vol. 27, No. 6, pp. 658-667, 1981.

31. Ellah, J. O., "Applications of Set Covering Theory to the Partitioning of Political Electoral Constituencies," Dissertation, University of Durham, England, 1981.

32. Wagner, W. H., "An Application of Integer Programming to Legislative Redistricting," CROND, Inc., Presented at the 34th National Meeting of ORSA, Philadelphia, Pennsylvania, November 1968.

33. Garfinkel, R. S. and Nemhauser, G. L., "Optimal Political Districting By Implicit Enumeration Techniques," Management Science, Vol. 16, No. 8, pp. 495-508, 1970.

34. Day, R. H., "On Optimal Extracting From A Multiple File Data Storage System: An Application of Integer Programming," Operations Research, Vol. 13, No. 3, pp. 482-494, 1965.

35. Nisijo, K. and Maruoka, A., "On Optimum Decomposition of Files," Trans. Inst. Electron. and Commun. Eng. Jpn. Sect. E., Vol. E62, No. 8, pp. 579-580, August 1979.

36. Fridshal, R. and North, J. H., "An Application of Linear Programming to the Minimization of Boolean Functions," Research Report, RC-472, IBM Research Center, June 1961.

37. Balinski, M. L., "Integer Programming: Methods, Uses, Computation," Management Science, Vol. 12, No. 12, pp. 253-313, November 1965.

38. Cobham, A., "A Statistical Study of the Minimization of Boolean Functions Using Integer Programming," Research Report, RC-756, IBM Research Center, June 1962.

39. Paul, M. C. and Unger, S. H., "Minimizing the Number of States in Incompletely Specified Sequential Functions," IRE Transactions on Electronic Computers, EC-8, pp. 356-367, 1959.

40. Root, J. C., "An Application of Symbolic Logic to a Selection Problem," Operations Research, Vol. 12, No. 4, pp. 519-526, 1964.

41. "Pattern Sequencing and Matching in Stock Cutting Operations," Tappi, Vol. 53, No. 4, pp. 668-678, April 1970.

42. Salveson, M. E., "The Assembly Line Balancing Problem," Journal of Industrial Engineering, Vol. 6, No. 3, pp. 18-25, 1955.

43. Steinmann, H. and Schwinn, R., "Computational Experience with a Zero-One Programming Problem," Operational Research, Vol. 17, No. 5, pp. 917-920, 1969.

44. Thuve, H., "Frequency Planning as a Set Partitioning Problem," European Journal of Operational Research (Netherlands), Vol. 6, No. 1, pp. 29-37, January 1981.

45. Morefield, C. L., "Application of 0-1 Integer Programming to Multi-target Tracking Problems," IEEE Trans. Autom. Control, Vol. AC-22, No. 3, pp. 302-312, June 1977.

46. Bessiere, F., "Sur la Recherce du Nombre Chromatique d'un Graphe par un Programme Lineaire en Nombres Entiers," Rev. Franc Recherche Operationelle, Vol. 9, pp. 143-148, 1965.

47. Busacher, R. G. and Saaty, T. L, Finite Graphs and Networks, McGraw-Hill, New York, 1965.

48. Cameron, S. H., "The Solution of the Graph Coloring Problem as a Set Covering Problem," IEEE Trans. Electromagn. Compat., Vol. 19, No. 3, pp. 320-322, 1977.

49. Bellmore, M., Greenberg, H. J., and Jarvis, J. J., "Multicommodity Disconnecting Sets," Management Science, Vol. 16, pp. B427-B433, 1970.

50. Aneha, Y. P. and Vemuganti, R. R., "Row Generation Scheme For Finding A Multi-Commodity Minimum Disconnecting Set," Management Science, Vol. 23, No. 6, pp. 652-659, 1977.

51. Bartholdi, III, J. J., Orlin, J. B., and Ratliff, H. D., "Cyclic Scheduling Via Integer Programs With Circular Ones," Operations Research, Vol. 28, No. 5, pp. 1074-1085, 1980.

52. Bartholdi, III, J. J., "A Guaranteed-Accuracy Round-Off Algorithm for Cyclic Scheduling and Set Covering," Operations Research, Vol. 29, No. 3, pp. 501-510, 1981.

53. Fulkerson, D. R, Nemhauser, G. L., and Trotter, L. E., "Two Computationally Difficult Set Covering Problems That Arise in Computing the 1-Width of Incidence Matrices of Steiner Triple Systems," Mathematical Programming Study, Vol. 2, pp. 72-81, 1974.

54. Karp, R. M., "Reducibility Among Combinatorial Problems," Complexity of Computer Computations, pp. 85-103, 1972.

55. Etcheberry, J., "The Set-Covering Problem: A New Implicit Enumeration Algorithm," Operations Research, Vol. 25, No. 5, 1977.

56. Orchard-Hays, W., Advanced Linear-Programming Computing Techniques, McGraw-Hill, 1968.

57. Graves, G. W., "A Complete Constructive Algorithm for the General Mixed Linear Programming Problem," Naval Research Logistics Quarterly, Vol. 12, No. 1, March 1965.

58. Bradley, G. H., Brown, G. G., and Graves, G. W., "Design and Implementation of Large-Scale Primal Transshipment Algorithms," Management Science, Vol. 24, No. 1, September 1977.

59. Brown, G. G. and McBride, R. D., "Efficient Solution of Generalized Network Problems," paper given at the ORSA/TIMS Conference in Detroit, Michigan, 1982.

60. Glover, F., Hultz, J., Klingman, D., and Stutz, J., "Generalized Networks: A Fundamental Computer-Based Planning Tool," Management Science, Vol. 24, pp. 1209-1220, 1978.

61. Srinivasan, V. and Thompson, G. L., "Stopped Simplex Algorithm for the Set Partitioning Problem With Transportation-like Subproblems," presented at the ORSA/TIMS Conference in Boston, April 24, 1974.

62. Brown, G. G. and Graves, G. W., "Elastic Programming: A New Approach to Large-Scale Mixed-Integer Optimization," paper presented at the ORSA/TIMS meeting, Las Vagas, Nevada, 17 November 1975.

63. Koene, J., "Processing Networks: Introduction and Basis Structure," Memorandum COSOR 81-06, Eindhoven University of Technology, Eindhoven, March 1981.

64. Hultz, J. and Klingman, D., "Solving Constrained Generalized Network Problems," Center for Cubernetic Studies, Research Report CCS 257, November 1976.

65. Klingman, D. and Russel, R., "Solving Constrained Transportation Problems," Operations Research, Vol. 23, No. pp. 91-108, 1975.

66. McBride, R. D., "Solving Generalized Network Problems With Side Constraints," Working Paper, School of Business Administration, University of Southern California, April, 1981.

67. Brown, G. G. and Wright, W. G., "Automatic Identification of Embedded Structure in Large-Scale Optimization Models," Large-Scale Linear Programming, Vol. 2, Proceedings of a IIASA Workshop, 2-6 June, 1980.

68. Brown, G. G., "Issues in Basis Manipulation: Factorization/Decomposition," The New Generation of L.P. Codes, paper presented at the ORSA/TIMS meeting, Miami, Florida, 5 November 1976.

69. Graves, G. W. and McBride, R. D., "The Factorization Approach to Large-Scale Linear Programming," Mathematical Programming, February 1976.

70. McBride, R. D., "Factorization in Large-Scale Linear Programming," Ph.D. Dissertation, UCLA, 1973.

71. Garfinkel, L. S. and Nemhauser, G. L., Integer Programming, Wiley, 1972.

72. Baker, E. K., "Efficient Heuristic Algorithms for the Weighted Set Covering Problem." Computers and Operations Research, Vol. 8, No. 4, pp. 303-310, 1981.

73. Brown, G. and Graves, G., XS Mathematical Programming System, perpetual working paper, (c. 1982).

74. International Business Machines Report SH20-0968-1, Mathematical Programming System--Extended (MPSX), and Generalized Upper Bounding, IBM Corporation, New York, 1974.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, CA 93940	2
3.	Department Chairman, Code 55 Department of Operations Research Naval Postgraduate School Monterey, CA 93940	1
4.	Professor Gerald G. Brown Code 55bw Department of Operations Research Naval Postgraduate School Monterey, CA 93940	100
5.	Captain Dan O. Bausch, USMC Headquarters Marine Corps (LPP) Washington, D.C.	2
6.	Commandant of the Marine Corps (Code RDRS-40) Headquarters United States Marine Corps Washington, D.C. 20380	2